Introduction to Machine Learning

JJ Vie

2020

Machine Learning

- Teach machine to learn from examples

Supervised learning (TP 1) & deep learning (TP 4)

Handwritten digit recognition, face recognition, etc.

Unsupervised learning (TP 2)

Recommender systems, clustering

Reinforcement learning (TP 3)

Robotics, games

TP 1: Supervised learning

From features $\boldsymbol{x}$ learn outcome y

Regression: continuous outcome

Ex. $y \in \mathbb{R}$ $\quad \mathcal{L} = \sum_i (\hat{y}_i - y_i)^2$ squared error

Classification: discrete outcome

Ex. $y \in \{0, 1\}$

- Logistic regression $\hat{y} = \sigma(\boldsymbol{w}^T \boldsymbol{x})$
- SVM classifier

Stochastic gradient descent

$\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma \nabla_{\boldsymbol{w}} \mathcal{L} \rightarrow$ gradient computed on a batch of data

```
from autograd import grad
parameters -= GAMMA * grad(loss)(parameters)
```

TP 2: Unsupervised learning

Learn $\boldsymbol{x}$

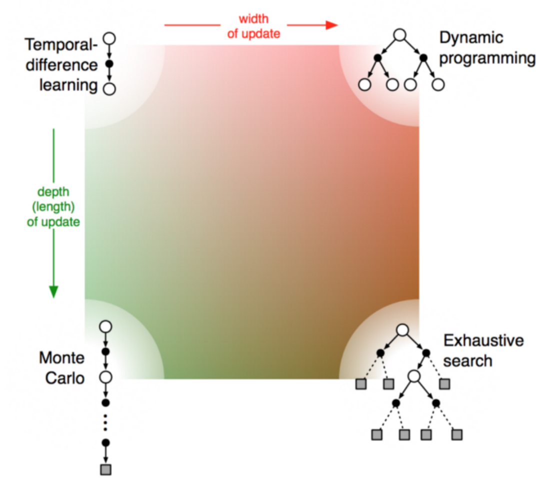
- K-nearest neighbors
- Dimensionality reduction (TP 3)
- Latent factor model (didn't have time)

TP 3: Reinforcement learning

Dynamic programming's *Principle of Optimality* (Bellman, 1952)

$$\left.\begin{array}{rr} \text{value function} & {\color{red}v_\pi(s)} \\ \text{action-value function} & {\color{red}q_\pi(s,a)} \end{array}\right\} \text{for policy } \pi : S \to A \to R \to S' \to A'$$

Q-learning (1992), SARSA (1996), policy gradient (2000)



TP 4: Deep learning

- Architecture of layers (CNN, dense) from input to output
- Ex. dense $n \to d_1 \to d_2 \to \ldots \to 1$

Deep neural networks are universal function approximators.

History

- 1958: Perceptron (Rosenblatt)
- 2012: ImageNet recognition (Krizkhevsky, Sutskever & Hinton)
- 2014: GANs image generation (Goodfellow et al.)
- 2016: AlphaGo beats Go world champion (DeepMind)
- 2016: WaveNet speech synthesis (DeepMind)
- 2017: AlphaGo Zero

Project: Roomba

Choose features $\boldsymbol{x}(S, A)$

Simplest (position), or tile coding

Choose function approximation

$Q(S, A) = \boldsymbol{w}^T \boldsymbol{x}(S, A)$ or $DeepNet(\boldsymbol{x}(S, A))$

Choose training algorithm

- Try a random policy, pick random $\boldsymbol{w}$
- SARSA with tile coding $\boldsymbol{x}(S, A)$
- Policy gradient?